Cross Site Location Jacking (XSLJ) (not really)

sirdarckcat and thornmaker

http://twitter.com/sirdarckcat
http://twitter.com/thornmaker
Fun With Redirects

sirdarckcat and thornmaker

http://twitter.com/sirdarckcat
http://twitter.com/thornmaker

OWASP
23.June.2010
About us...

- Eduardo Vela Nava (sirdarckcat)

Enjoys

- Making up absurd names for presentation titles
- Managing IBOS (International Buzzword Organization for Security)
- Hacking on anything produced by google|microsoft
About us...

- David Lindsay (thornmaker)
- Enjoys
  - TV shows about likeable serial killers
  - Finnish chocolate
  - Finnish sauna

- Works for Cigital Inc
  - Offices in USA, England, and Amsterdam
  - And yes, we're hiring :)
Redirects

- 300 Multiple Choice
- 301 Moved Permanently
- 302 Found
- 303 See Other
- 307 Temporary Redirect
Location Header

- Contains destination of redirect
- Location: [http://example.org](http://example.org)
- Cannot redirect to javascript:

- That's all, right?

- Nope...
Refresh

- Refresh: 0; url=http://example.org

- The initial 0 is the time delay before redirection

- Works with status code 200, and many others
Meta Redirects

- `<meta http-equiv="Refresh" content="0; url=http://www.example.com/" />`

- Location header redirect always trumps
JavaScript Redirects

- `window.open('http://0x.lv')`
- `location.replace('http://0x.lv')`
- `location.assign('http://0x.lv')`
- `location.href='http://0x.lv/'`
- `location='http://0x.v/'`
- `location.port='8080' //sorta`
- `etc...`
- `document.URL` (IE only)
- `URL` (in event handlers, IE only)
Others methods

- Flash
  - LoadVars().send()
  - getURL()
  - etc
- PDFs
- Java
- Special URI handlers
- and more
Owasp Top 10

- 2010 version of Owasp Top 10
- "Attacker links to unvalidated redirect and tricks victims into clicking it."

- Unvalidated redirect?
- [http://example.com/redirect?url=0x.lv](http://example.com/redirect?url=0x.lv)
Open Redirects

Security Problem?

Yes!
- They enable phishing/malware.
- Make browser/plugin vulnerabilities exploitable.
- Break trust on whitelists of URLs for resources.

No!
- If you take care of phishing/malware.
- If you decide to require browser/plugin vendors to fix vulns.
- If you decide not to trust, and tell everyone not to trust whitelists on your applications.
- It's hard.. very hard.
Open Redirects

Security Problem?

More or less
- You have to remember you have open redirects.
- You have to find an alternative for URL whitelists.
- You have to rely on the security of browser/plugin vendors.

Generally?
- You have to assume everyone has open redirects.
- You can't use URL whitelists most of the times.
- C'est la vie.
- You may as well just use them..
Open Redirects

Are open redirects ever useful?

Sometimes…
- Track user clicks/activities (a@ping didn't work).
- Handle complex session interaction (login/logout).
- Interrupt/modify navigation flow.
- etc..
Open Redirects

Solutions?

- Attempt #1: Signing/encrypting the URL to redirect.
- FAIL: If attacker can just let you sign it for them.

- Attempt #2: Check the URL, and verify who it belongs to
- FAIL: URLs aren't easy to parse, everyone does it differently:

Following demos and more available at: http://www.sirdarckcat.net/uritest.html
URL Parsing

URL Parsing is hard.

- Example 1 (fixed, found by WHK):

  How do you parse http:///evil.com/attack? (with 3 /)

| http: -> scheme               | http: -> scheme            |
|/// -> scheme-host separator  | // -> scheme-host separator|
|evil.com -> hostname          | -> hostname                |
|/ -> host-path separator      | / -> host-path separator   |
|attack -> path                | evil.com/attack -> path    |
URL Parsing

- Example 2 (unfixed, PHP):

We have: http://hostname/path/to/file.php
PHP_SELF = /path/to/file.php

```html
<a href='$PHP_SELF'>
</a>
```

We have: `<a href="/www.google.com">`  
Links to: http://www.google.com/

```html
<a href="/www.google.com/..//path/to/file.php">  
http://www.google.com/
</a>
```
How to parse URLs correctly?

Don't try to do it! (or at least be very careful when you do)

- Even if you get it right, browsers won't.

- Simple examples (all your answers will be wrong):
How to do it correctly?

What's the TLD?

http://facebook.com & .google.com & .yahoo.com

It depends!!!
How to do it correctly?

What's the hostname?

http://www.google.com/

When the URL is loaded at http://www.example.com/ then it will point to http://www.example.com/www.google.com

When the URL is loaded at https://ssl.example.com/ then it will point to http://www.google.com/ or to https://ssl.example.com/http://www.google.com (depending upon the browser)
How to do it correctly?

Which domain will be loaded?

http://google.com:paypal.com/

Firefox 3.5 and Opera will send you to google.com

Other browsers will give an error
URL Parsing

All exceptions we've found are each a different judgment call on an unexpected situation.

- URLs represent:
  - Relative links (to the current document? not really)
  - Absolute links (how to know if they are absolute?)

- People will tell you there are rules, don't believe them.

- RFC's are not as clear as they could be.

- HTML5 refers you to the unclear RFC's.

- Lot's of implementation differences.
Exceptions

Note the following sites allow redirects:

1. Search engines (google/bing/yahoo)
2. Some login sites (facebook/youtube)
3. OpenID customers/providers (almost all.. a few don't)
Conclusion..

- Don't trust hostname-based whitelists unless you are completely sure they don't have open redirects.

- Check how your URL parser behaves on several browsers.

- Redirects are a main component of HTTP functionality. We won't take them away, and they are used a lot.

- They are dangerous because of developers that forget about them.
Reminder

URLs are evil!

Even if you check that the URL you are loading is

• http://www.ponies.com/

It may end up redirecting to

• file://etc/shadow

URLs don't represent a resource, and they are not uniform..

Remember URLs as: Unfortunate Redirect Launchers
The content of this slide has been removed by request of Adobe.
The content of this slide has been removed by request of Adobe.
The content of this slide has been removed by request of Adobe.
The content of this slide has been removed by request of Adobe.
URL Shorteners - <rant>

- URL shorteners are EVIL! Why?
- Condition users to click links that take them to an unknown location

- [http://www.example.com/redirect?url=%68%74%74%70%3A%2F%2F%65%76%69%6C%77%65%62%73%69%74%65%2E%63%6F%6D%2F%70%77%6E%7A%2E%70%68%70](http://www.example.com/redirect?url=%68%74%74%70%3A%2F%2F%65%76%69%6C%77%65%62%73%69%74%65%2E%63%6F%6D%2F%70%77%6E%7A%2E%70%68%70)  <--- a bit suspicious still, no?
- [http://tinyurl.com/36lnj2a](http://tinyurl.com/36lnj2a)  <--- When was the last time you clicked on a link just like this?
URL Shorteners (rant continued)

■ Theory is one thing... what about real life?


■ Jira message (05.April.2010):
  ive got this error while browsing some projects in jira http://tinyurl.com/XXXXXXXXXX
URL Shorteners (rant continued)

What were the consequences?

- Clicking on tinyurl.com clink -> XSS
- XSS + Bruteforcing login -> Compromised JIRA admin account
- -> disable notifications
- -> change upload path
- -> upload JSP files
- -> copy user's home directories + backdoor access
- -> install jar file to collect logins + passwords
- -> use admin's password to access other server with root privileges
- -> use cached svn passwords to access other server
URL Shorteners (rant continued)

- Can URL shorteners be made more secure?
- Blacklisting destinations? um... no.
- Whitelisting destinations? better but no. wouldn't have helped apache.
- Request Policy (FF Extension): prompts on every redirect. Can be annoying but is configurable.
- Mandatory page preview e.g. http://tinyurl.com/preview.php
- </rant>
Reading Redirects

- If a page makes a request for a URL which is redirected, the launching page cannot access the destination URL.

- Why? The launching page could learn sensitive information such as login names, user IDs, authentication and authorization tokens (in the URL) and so forth.
Reading Redirects – First known example


- URL token stealing via stylesheet redirect

- ".href property of stylesheet DOM nodes [...] reflect the final URI of the stylesheet after following any 302 redirects"
Reading Redirects – Second example

- Cesar Cerrudo -

- Exact same issue with webkit (was fixed)

- "There are still similar redirect leak bugs floating around other browsers though. " – kuza55
Reading Redirects – Third example

- Soroush Dalili -
  [http://soroush.secproject.com/downloadable/XSUH_FF_1.pdf](http://soroush.secproject.com/downloadable/XSUH_FF_1.pdf) and
  [http://0me.me/demo/XSUH/XSUH_demo_firefox_all_in_1.html](http://0me.me/demo/XSUH/XSUH_demo_firefox_all_in_1.html)

- Uses the IBOS non-approved term XSUH (should be XSLJ because it has cross-site *and* jacking in it!)

- `<script src="http://www.yahoo.com">`
Reading Redirects – Latest to be released

- Eduardo Vela -

- Firefox only, same-origin policy bypass

- Referred to as XSLJ, making it officially IBOS compliant :)
Play Tool

- **http://0x.lv/xss.php?source**


- The tool was developed for XSS testing but is great for playing with redirection issues too :)
IBOS Work

- We are now accepting nominations for additional buzzwords to attach to the following issues:
  - XSS + Clickjacking
  - XSRF + HPP
  - SQLi + XSS
  - SJ + RFI
Thanks

- Thanks to AppSecEU committee for the drinks, the contests, and for the invitation :)

- Thanks to kuza55 (for you know what)

- Thanks to you all for attending!!!